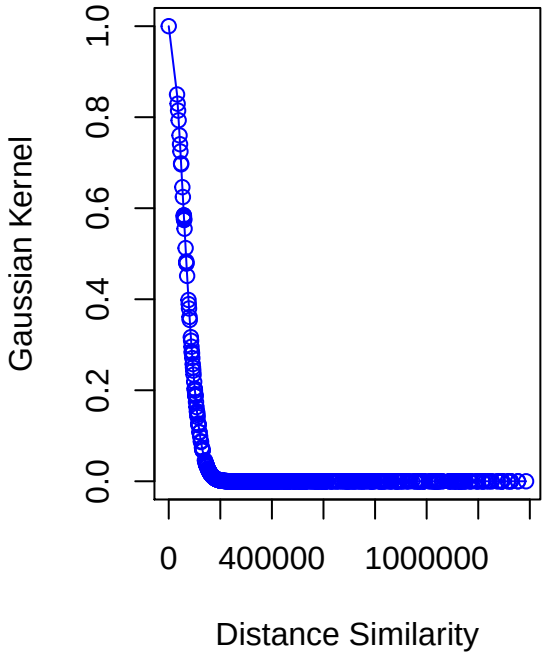
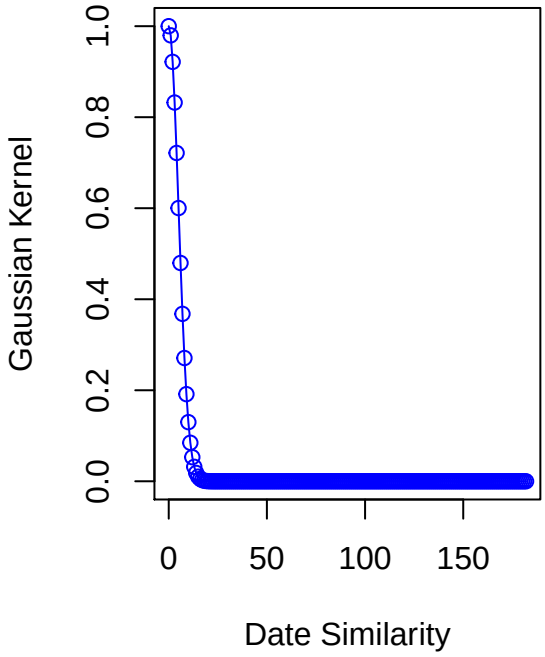
732A99 ComputerLab3 Block1

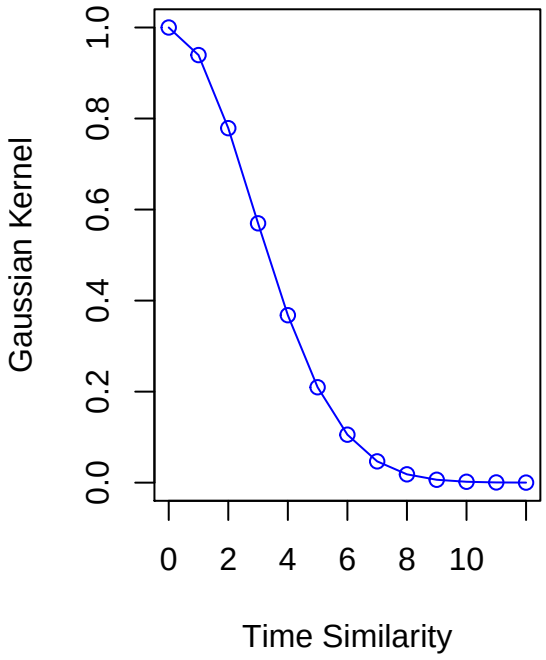
Namita Sharma

12/17/2019

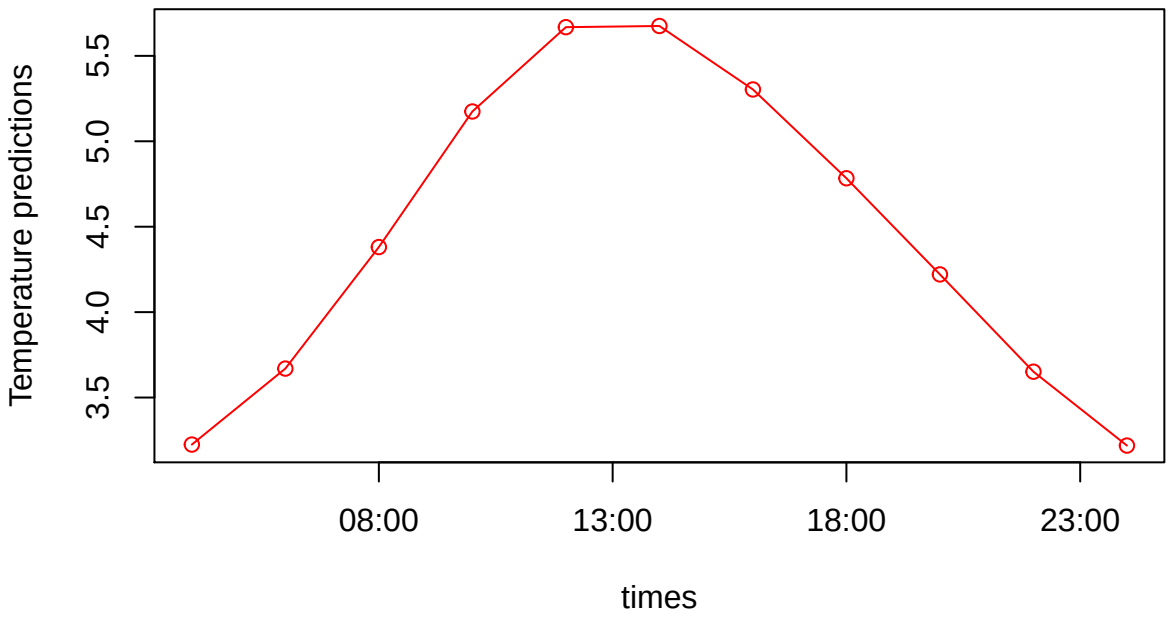
1. KERNEL METHODS

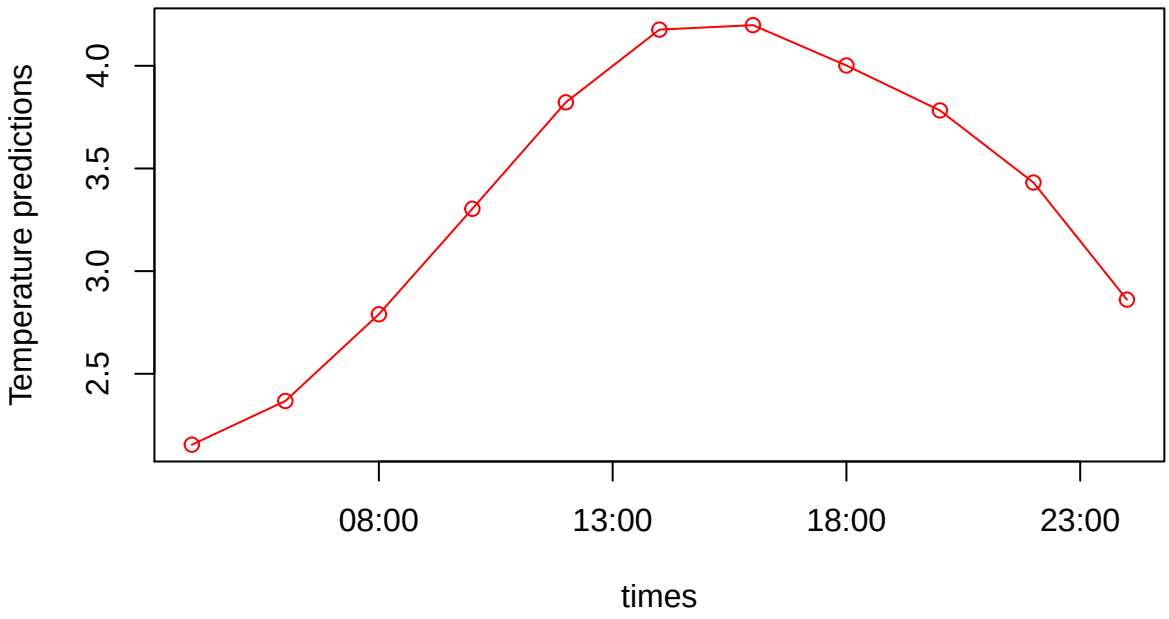






As can be seen from the three gaussian kernel plots for distance, date and time similarities, the widths are chosen according to the range of values in each distribution. As difference in distances are in the order of hundreds of miles, we choose "close" distances to be within 50 miles (80000 meters). Similary, all dates within one week (7 days) of the target date are considered to be close and given more weights here. And all times within a range of 4 hours from the target time will be given more say in the predictions.





The temperature predictions from the multiplied gaussian kernel varies a bit more than the temperature predictions from the additive gaussian kernel. In general, the multiplicative kernel has a much lesser width than the additive kernel, i.e. very few points from the data are weighted in the predictions.

2. SUPPORT VECTOR MACHINES

We can use the holdout method to determine which of the three models (C = 0.5, 1, 5) is the most suitable for classifying spam emails. Summary of each of the three models is given below.

	No.of.Support.Vectors	Cost.Parameter	Validation.Error
SVM model 1	1077	0.5	0.0860870
SVM model 2	1007	1.0	0.0695652
SVM model 3	966	5.0	0.0747826

We can see that SVM Model 2 with C = 1 is the most promising as it gives minimum classification error on the validation data. By using this model to predict on the unseen data (test set), the generalization error can be estimated.

Table 1: Test Confusion Matrix

	nonspam	spam
nonspam	643	31
spam	67	410

Generalization.Error
0.0851434

The optimum SVM model

```
## Support Vector Machine object of class "ksvm"
```

```
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.05
##
## Number of Support Vectors : 1007
##
## Objective Function Value : -467.7423
## Training error : 0.04
```

The parameter C defines the cost of constraints violation this is the 'C'-constant of the regularization term in the Lagrange formulation. Increasing the C values imposes higher penalties for misclassifications. It can be seen that greater the value of C, fewer support vectors are used in the model.

Appendix

```
#########################################################################
# Assignment 1. Kernel Methods
#########################################################################
library("geosphere")
library("hms")

# Import station data
stations <- read.csv(
  file   = "C:/Users/namit/Downloads/Machine Learning/Lab3 Block1/stations.csv",
  header = TRUE)

# Import temperature data
temps <- read.csv(
  file   = "C:/Users/namit/Downloads/Machine Learning/Lab3 Block1/temps50k.csv",
  header = TRUE)

set.seed(1234567890)
st <- merge(stations, temps, by = "station_number")

# Format Date and Time data
st$date <- as.Date(st$date)
st$time <- strptime(st$time, "%H")

# Paramters of interest
lat    <- 58.4274                                          # Point to predict - latitude
long   <- 14.826                                           # Point to predict - longitude
date   <- "2013-11-04"                                     # Date to predict
times  <- c("04:00:00", "06:00:00", "08:00:00", "10:00:00",   # Times to predict
            "12:00:00", "14:00:00", "16:00:00", "18:00:00",
            "20:00:00", "22:00:00", "24:00:00")

# Format date and time and filter future temperature measurements
date      <- as.Date(date)
```

```r
times     <- strptime(times, "%H")
st_subset <- st[st$date < date, ]

# Distance similarity from a station to the point of interest
distance.sim <- distHaversine(c(lat, long), st_subset[, c("latitude", "longitude")])

# Distance similarity between the day of temperature measurement and day of interest
date.sim     <- difftime(date, st_subset$date, units = "days")
date.sim     <- sapply(as.numeric(date.sim), function(x) {
  min(x %% 365, 365 - (x %% 365))
})

# Distance similarity between hour of temperature measurement and hour of interest
time.sim <- sapply(times, function(x) {
  timediff <- difftime(x, st_subset$time, units = "hours")
  timediff <- sapply(abs(as.numeric(timediff)), function(y) {
    min(y, 24 - y)
  })
  return(timediff)
})

# Width of gaussian kernels
h_distance <- 80000  # 50 miles
h_date     <- 7      # 1 week
h_time     <- 4      # 4 hours

# Gaussian kernels for distance, date and time similarity
k_distance <- exp(-1 * (distance.sim / h_distance) ^ 2)
k_date     <- exp(-1 * (date.sim / h_date) ^ 2)
k_time     <- exp(-1 * (time.sim / h_time) ^ 2)

# Plots of the three gaussian kernels
layout(mat = matrix(c(1:2), 1, 2, byrow = TRUE))

dist.order <- distance.sim[order(distance.sim)]
dist.gauss <- exp(-1 * (dist.order / h_distance) ^ 2)
plot(x    = unique(dist.order),
     y    = unique(dist.gauss),
     xlab = "Distance Similarity",
     ylab = "Gaussian Kernel",
     type = "o",
     col  = "blue")

date.order <- date.sim[order(date.sim)]
date.gauss <- exp(-1 * (date.order / h_date) ^ 2)
plot(x    = unique(date.order),
     y    = unique(date.gauss),
     xlab = "Date Similarity",
     ylab = "Gaussian Kernel",
     type = "o",
     col  = "blue")

time.order <- time.sim[order(time.sim[, 1]), 1]
```

```r
time.gauss <- exp(-1 * (time.order / h_time) ^ 2)
plot(x    = unique(time.order),
     y    = unique(time.gauss),
     xlab = "Time Similarity",
     ylab = "Gaussian Kernel",
     type = "o",
     col  = "blue")
#-------------------------------------------------------------------------
# Sum of three gaussian kernels
k_gauss_sum <- k_distance + k_date + k_time

# Temperature predictions
temp.pred1 <- sapply(seq_along(1:ncol(k_gauss_sum)), function(i) {
  sum(k_gauss_sum[, i] * st_subset$air_temperature) / sum(k_gauss_sum[, i])
})

# Plot temperature predictions
plot(x    = times,
     y    = temp.pred1,
     ylab = "Temperature predictions",
     type = "o",
     col  = "red")

#-------------------------------------------------------------------------
# Product of three gaussian kernels
k_gauss_prod <- k_distance * k_date * k_time

# Temperature predictions
temp.pred2 <- sapply(seq_along(1:ncol(k_gauss_prod)), function(i) {
  sum(k_gauss_prod[, i] * st_subset$air_temperature) / sum(k_gauss_prod[, i])
})

# Plot temperature predictions
plot(x    = times,
     y    = temp.pred2,
     ylab = "Temperature predictions",
     type = "o",
     col  = "red")

######################################################################
# Assignment 2. Support Vector Machines
######################################################################
library("kernlab")

# Load spam data
data(spam)

# No of observations in the spam dataset
n <- dim(spam)[1]

# Divide dataset into training, validation and test data (50%, 25%, 25%)
RNGversion('3.5.1')
set.seed(12345)
```

```r
id1    = sample(1:n, floor(n / 2))
train  = spam[id1, ]
id_rem = setdiff(1:n, id1)

set.seed(12345)
id2   = sample(id_rem, floor(n / 4))
valid = spam[id2, ]
id3   = setdiff(id_rem, id2)
test  = spam[id3, ]

# Three SVM models with C values 0.5, 1 and 5
j <- 1
svm_model <- list()
cfmat     <- list()
mc_rate   <- numeric()

for(i in c(0.5, 1, 5)) {
  # SVM model with different C values
  svm_model[[j]] <- ksvm(x       = type ~ .,
                         data    = train,
                         type    = "C-svc",
                         C       = i,
                         kernel  = "rbfdot",
                         kpar    = list(sigma = 0.05))

  valid.pred <- predict(object  = svm_model[[j]],
                        newdata = valid,
                        type    = "response")

  cfmat[[j]] <- table(actual    = valid$type,
                      predicted = valid.pred,
                      dnn       = c("Truth", "Prediction"))

  mc_rate[j] <- 1 - (sum(diag(cfmat[[j]])) / sum(cfmat[[j]]))

  j <- j + 1
}
# Generalization Error
test.pred  <- predict(object  = svm_model[[2]],
                      newdata = test,
                      type    = "response")

cfmat_test <- table(actual    = test$type,
                    predicted = test.pred,
                    dnn       = c("Truth", "Prediction"))

mc_test    <- 1 - (sum(diag(cfmat_test)) / sum(cfmat_test))
# SVM for user
print(svm_model[[2]])
```